Super Python Talks for Life Sciences

Statistics using `scipy.stats`

Ajayrama Kumaraswamy

Dept. Biologie II, Biozentrum der LMU, Martinsried

22^{nd} Feb 2017

Jupyter Notebook Intro and Demo

From `jupyter.org`:

> *The Jupyter Notebook is a web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more.*

Demo

Matplotlib inline Demo

Outline

Using `scipy.stats`,

Random Variables and Distribution Functions

- Plotting Distribution Functions
- Making Custom Random variables, moments and drawing samples

Analyzing One Sample

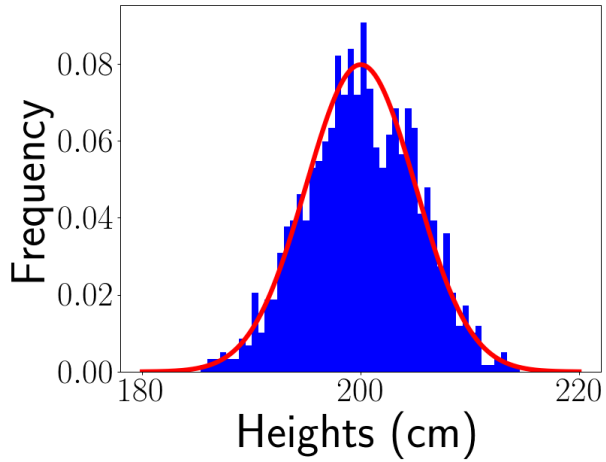
- First look: stats, histogram
- Kernal Density estimation and distribution fitting
- Significance tests: t-test and Kolmogorov-Smrinov Test
- Special tests normality

Analyzing two samples

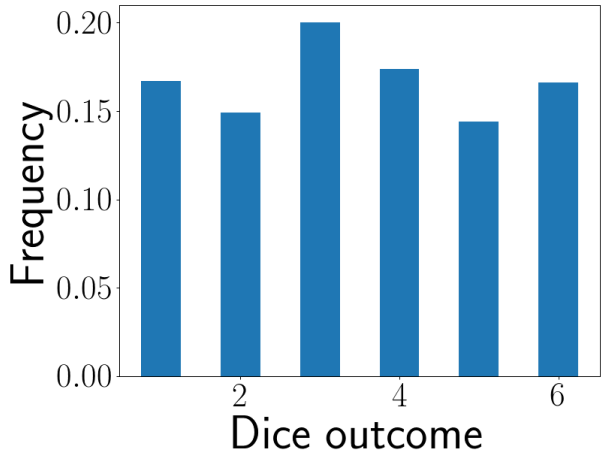
- Significance test

Simple Introduction to Random Variables and Distribution Functions

Measuring heights of 100 people



Rolling a Die



Implementation in `scipy.stats`

Random Varibles as Classes

Continuous

- `scipy.stats.norm`
- `scipy.stats.gamma`
- `scipy.stats.expon`

Discrete

- `scipy.stats.bernoulli`
- `scipy.stats.binom`
- `scipy.stats.poisson`

Distribution Functions as static/instance methods of classes

Continuous

- `norm.pdf`
- `norm.cdf`
- `norm.ppf`
- `norm.sf`
- `norm.isf`

Discrete

- `bernoulli.pmf`
- `berboulli.cdf`
- `bernoulli.ppf`
- `bernoulli.sf`
- `bernoulli.isf`

See "Specific Points for Discrete Distributions" on scipy-ref-0.18.1.pdf pg.224

Specifying RVs in `scipy.stats`

- Form/shape of associated distributions
- Formula and parameters
- Default parameters: location and scale
- Shape parameters for some distributions

Demo

Plotting PDF and CDF

Demo

Custom Random Variables

`stats.rv_continuous`

Subclass and define either PDF or CDF, normalizing such that `loc`=0 and `scale`=1

```
1 class deterministic_gen(stats.rv_continuous):
2     def _cdf(self, x):
3         return np.where(x < 0, 0., 1.)
```

`stats.rv_discrete`

Subclass and define PMF, normalizing such that `loc`=0 and `scale`=1

```
1 class poisson_gen(rv_discrete):
2 "Poisson distribution"
3     def _pmf(self, k, mu):
4         return exp(-mu) * mu**k / factorial(k)
```

If required `_argparse` and other functions

Demo

Drawing Samples

```
1 stats.norm.rvs(<shape parameters>, size=<>, loc=<>,
      scale=<>)
```

- size must not be omitted, otherwise considered as shape paramter
- set `np.random.seed` before drawing to get replicable samples

Other Statical Functions

```
1 stats.norm.mean(<shape parameters>, loc=<>, scale
      =<>)
2 stats.norm.median(<shape parameters>, loc=<>, scale
      =<>)
3 stats.norm.std(<shape parameters>, loc=<>, scale
      =<>)
4 stats.norm.var(<shape parameters>, loc=<>, scale
      =<>)
5 stats.norm.moment(n, <shape parameters>, loc=<>,
      scale=<>)
6 stats.norm.stats(<shape parameters>, moments=<
      string of 'm', 'v', 's', 'k'>,loc=<>, scale=<>)
```

- 'm': mean
- 'v': variance
- 's': skewness
- 'k': kurtosis

Demo

Analysis of One Sample: Statistical Functions

- `np.mean`, `np.std`, `np.var`, `np.median`
- `stats.describe`: return nobs, mean, variance, skewness, kurtosis
- `stats.mode`, `stats.skew`, `stats.kurtosis`, `stats.moment`, `stats.sem`
- `stats.gmean`: geometric mean; nth root of $(x_1 * x_2 * \cdots * x_n)$
- `stats.hmean`: harmonic mean; $n/((1/x_1) + (1/x_2) + \cdots + (1/x_n)$

Trimmed Functions and Binned Statistic

Trimmed Statistics

Only use samples with a certain bounds

`stats.tmin`, `stats.tmax`, `stats.tmean`, `stats.tstd`, `stats.tvar`

Binned statistic

Generalized version of the histgram. More in Demo.

Demo

Kernal Density Estimation and Distribution Fitting

Kernal Desity Estimation

`stats.gaussin_kde(dataset=, bw_method=)`

Distribution Fitting

`stats.gamma.fit(data, *args, **kwargs)`

- *args: starting values of any shape parameters
- **kwargs: starting values (Ex: loc=1, scale=2)
  or fixed values of parameters(Ex: floc=2, fscale=1, fa=2, f0=2)

Demo

Significance Tests

One sample

- `stats.ttest_1samp`
- `stats.kstest`
- `stats.normaltest`

Two Sample

- `stats.ttest_ind`
- `stats.ks_2samp`
- `f_oneway`
- `pearsonr`
- `spearmanr`
- And many more...

`http://www.biostathandbook.com/`

Review

Reference Manual

`https://docs.scipy.org/doc/scipy/reference/stats.html`

Scipy Tutorial

`https://docs.scipy.org/doc/scipy/reference/tutorial/stats.html`

Thanks!

- to you all!
- Nick Del Grosso
- Members of Wachtler lab.

Questions?